IN THE CLAIMS

Please amend claims 1, 10, 12-14, and 28 as follows:

1. (Currently Amended) A computer-implemented method for rendering graphics on an embedded device, comprising:

inputting rendering data in a first format;

converting the rendering data from the first format into a variable length fixed-point format in a normalized homogeneous coordinate system;

processing the rendering data in the variable-length fixed-point format; and

rendering the processed rendering data on the embedded device.

2. (Original) The computer-implemented method of claim 1, wherein processing the rendering data further comprises using a normalized homogenous coordinate system (NHCS) for vector operations on the rendering data.

3. (Original) The computer-implemented method of claim 2, wherein the first format is at least one of: (a) floating-point format; (b) fixed-point format.

4. (Original) The computer-implemented method of claim 1, further comprising creating a mathematical library for processing the rendering data in a variable-length fixed-point format.

5. (Original) The computer-implemented method of claim 4, processing the rendering data further comprises performing fixed-point mathematical operations contained in the mathematical library on the rendering data.

6. (Original) The computer-implemented method of claim 4, wherein processing the rendering data further comprises computing graphic functions contained in the mathematical library using the rendering data.

7. (Original) The computer-implemented method of claim 2, further comprising predicting a range of the processed rendering data and truncating any data outside the range.

8. (Original) The computer-implemented method of claim 1, wherein the embedded device includes a mobile computing device using Direct3D for mobile devices.

9. (Original) A computer-readable medium having computer-executable instructions for performing the computer-implemented method recited in claim 1.

10. (Currently Amended) A process for rendering graphics on an embedded computing platform, comprising:

inputting rendering data;

converting the rendering data into a variable-length fixed-point format including a normalized homogenous coordinate system (NHCS) for vector operations;

defining a data structure for the converted rendering data to generate converted rendering data in a NHCS fixed-point format;

using a fixed-point mathematical library to process the NHCS fixed-point format rendering data; and

rendering the processed NHCS fixed-point format rendering data on the embedded computing platform.

11. (Original) The process as set forth in claim 10, wherein the fixed-point mathematical library includes mathematical operations and graphics functions in an NHCS fixed-point format.

12. (Currently Amended) The process as set forth in claim 10, further comprising predicting a range of the processed NHCS fixed-point format rendering data.

13. (Currently Amended) The process as set forth in claim 12, further comprising truncating any processed NHCS fixed-point format rendering data outside of the predicted range.

14. (Currently Amended) The process as set forth in claim 10, wherein rendering the processed NHCS fixed-point format rendering data further comprises using Direct3D mobile (D3DM).

15. (Original) The process as set forth in claim 14, wherein the Direct3D mobile (D3DM) uses the NHCS fixed-point format to represent the rendering data instead of a floating-point representation.

16. (Original) The process as set forth in claim 10, wherein inputting rendering data further comprises inputting rendering data in at least one of the following formats: (a) floating-point format; (2) fixed-point format.

17. (Original) One or more computer-readable media having computer-readable instructions thereon which, when executed by one or more processors, cause the one or more processors to implement the process of claim 10.

18. (Original) A computer-readable medium having computer-executable instructions for preparing data for rendering on a computing device, comprising:

converting the data into a variable-length fixed-point format having a normalized homogenous coordinate system (NHCS) to generate NHCS fixed-point data;

creating specialized buffers on the computing device to store the NHCS fixed-point data;

processing the NHCS fixed-point data using a mathematical library capable of computing mathematical operations and graphics functions using a NHCS fixed-point format; and

preparing the processed NHCS fixed-point data for raster by translating the NHCS fixed-point data into a language of the computing device's graphics hardware.

19. (Original) The computer-readable medium of claim 18, further comprising inputting the data in a floating-point format.

20. (Original) The computer-readable medium of claim 18, further comprising inputting the data in a fixed-point format.

21. (Original) The computer-readable medium of claim 18, wherein preparing the processed NHCS fixed-point data for raster by further comprises converting 3D coordinates of the NHCS fixed-point data into 2D screen coordinates.

22. (Original) The computer-readable medium of claim 18, further comprising using a Direct3D for mobile (D3DM) rendering standard to render the NHCS fixed-point data on the computing device, wherein the D3DM rendering standard accepts data in the NHCS fixed-point format instead of a floating-point format.

23. (Previously Presented) A method for converting a format of rendering data, comprising:

inputting the rendering data in at least one of the following formats: (a) floating-point format; (b) fixed-point format;

identifying a maximum value in the rendering data; and

normalizing remaining values in the rendering data based on the maximum value to generate the rendering data in a normalized homogenous coordinate system (NHCS) variable length fixed-point format.

24. (Original) The method as set forth in claim 23, further comprising determining a maximum fixed-point buffer size of a destination buffer.

25. (Original) The method as set forth in claim 24, further comprising scaling the maximum value to the maximum fixed-point buffer size.

26. (Original) The method as set forth in claim 25, further comprising recording a shift digit value used in the scaling.

27. (Original) The method as set forth in claim 26, wherein normalizing further comprising using the shift digit to normalize the remaining values.

28. (Currently Amended) A graphics rendering system for an embedded computing device, comprising:

a task module that inputs raw rendering data in a first format and converts the raw rendering data into a second format that is a variable-length fixed-point format in a normalized homogeneous coordinate system;

an application programming interface (API) module that creates buffers for storing the converted rendering data;

a driver module that processes the converted rendering data to prepare the converted rendering data for rendering; and

a rendering engine that renders the processed rendering data on the embedded computing device.

29. (Previously Presented) The graphics rendering system as set forth in claim 28, wherein the variable-length fixed-point format is a normalized homogenous coordinate system (NHCS) such that the rendering data is in a NHCS fixed-point format.

30. (Original) The graphics rendering system as set forth in claim 29, wherein the first format is at least one of: (a) floating-point format; (b) fixed-point format.

31. (Original) The graphics rendering system as set forth in claim 28, wherein the task module further comprises a math library and translator that converts the raw

rendering data and performs preliminary mathematical operations on the raw rendering data.

32. (Original) The graphics rendering system as set forth in claim 28, wherein the API module further comprises an index buffer that stores indices.

33. (Original) The graphics rendering system as set forth in claim 28, wherein the API module further comprises a vertex buffer that stores vertex information.

34. (Original) The graphics rendering system as set forth in claim 28, wherein the API module further comprises a wrapper that packages commands and provides convenience, compatibility and security for the commands.

35. (Original) The graphics rendering system as set forth in claim 34, wherein the API module further comprises a command buffer that stores the wrapper.

36. (Original) The graphics rendering system as set forth in claim 28, wherein the task module further comprises a transform and lighting module that prepares the converted rendering data for a rasterizer.

37. (Previously Presented) The graphics rendering system as set forth in claim 36, wherein the transform and lighting module further comprises a fixed-point mathematical library that processes the converted rendering data in a fixed-point format.